Final Report

NESSUS/NASTRAN Interface
(Modification of NESSUS to FORTRAN 90 Standard)

SwRI Project 06-7212

NASA Project NAS8-39797/mod 4

Objective

The objective of this work has been to develop a FORTRAN 90 (F90) version of the NESSUS probabilistic analysis software, Version 6.2 with NASTRAN interface. The target platform for the modified NESSUS code is the SGI workstation.

Work Performed

There were two major issues to be addressed in porting NESSUS 6.2 to F90 on the SGI.

1) Integer manipulation of character data
2) Hollerith descriptors in FORMAT statements.

Integer manipulation of character data

The biggest obstacle to overcome in porting NESSUS to FORTRAN 90 concerned the reading, manipulation and writing of character data within NESSUS/FEM. The NESSUS 6.2 code delivered in September 1995, reads character values into integer arrays, manipulates the integer values, e.g., performs keyword comparisons, and writes integer variables using an "A" format. This approach is legal under FORTRAN 66 and FORTRAN 77 but not FORTRAN 90. Some F90 compilers, e.g., HP, allow this but SGI does not. Also, NESSUS initializes integers in data statements as characters using Hollerith statements.

After evaluation of the NESSUS code, it was determined that some routines could be fixed easily by changing the array from integer to character. This was possible in routines where the array in question was a local array.

However, other arrays were passed as arguments to a large number of subroutines. Any significant change to these arrays would effect the entire NESSUS/FEM reading procedure. Therefore, a work around was developed.

For example, the FORTRAN code below is used to read a character and determine whether it is an '*' or not. The code, 1) initializes an integer, `istar`, to a character '*' in a data statement, then, 2) reads data into an integer array, then, 3) compares the integers.

```
      data istar / 1H* /

      read(icread,100) ichar
100   format(a1)

      if (ichar.eq.istar) go to 200
```

Under FORTRAN 90 on the SGI, the data statement and the read statement are illegal.

The approach taken within NESSUS/FEM to circumvent this problem is to read the data as characters and transfer the information to integers. Then, NESSUS/FEM can use the same integer-based character operations as before. However, the integers must be changed back to characters again before printing.

FORTRAN 90 has a `TRANSFER` intrinsic that can be used to transfer between character and integer. `TRANSFER` does not adjust the bits of a variable, but changes the interpretation of the bits. `TRANSFER` is a function with syntax

```
TRANSFER(variable_to_transfer, type_to_transfer_to).
```

Therefore, we can use the code

```
      istar = TRANSFER('*   ',istar)

      read(icread,100) char
100   format(a1)

      ichar = TRANSFER(char//'    ',ichar)

      if (ichar.eq.istar) go to 200
```

to set `istar` and `ichar`. Then, the integer comparison code can be used without changes.

Note that a character is 1 byte whereas an integer is 4 bytes. Thus, the character variable `char` is padded by three blanks before transferring to an integer.

The code can be optimized by placing the initialization of an integer variable into a data statement to prevent the repeated setting of `istar`. This can be done by identifying the resulting integer code for character variables and placing the values in parameter statements. The parameters can then be used to set the integer variables. The parameters are contained in a

module "f90_params.f90". f90_params.f90 is listed in Appendix A.

The code can now be written as (where STF90 is the integer code for '* ')

```
      use f90_params
      data istar / STF90 /

      read(icread,100) char
100   format(a1)

      ichar = TRANSFER(char//'    ',ichar)

      if (ichar.eq.istar) go to 200
```

This approach works correctly. The difficulty is that numerous changes are required throughout NESSUS.

A related issue is the writing of character data. Writing integer variables as characters is allowable under FORTRAN 77, but not FORTRAN 90 on the SGI. Thus, those statements must also be modified by using a temporary character variable corresponding to the integer to be printed. For example, the code

```
      write(iout,300) ichar
300   format(a1)
```

is allowable under FORTRAN 77. An equivalent FORTRAN 90 code is

```
      char = transfer(ichar,chr)
      write(iout,300) char(1:1)
300   format(a1)
```

where char is declared character*4.

Hollerith descriptors within FORMAT statements

Several FORMAT statements contained Hollerith descriptors, which are marked for obsolescence in F90 and are not allowed on the SGI. This fix was to replace all Hollerith descriptors with apostrophe delimiters.

Modified subroutines

A list of the subroutines modified and the particular change for each routine is given in Appendix B.

SGI Source Code, Executable and Verification Problems

The modified NESSUS source code and verification problems are located on the MSFC SGI with Internet address `cosmo.msfc.nasa.gov` in the following directory structure:

1. hmill/f90
   a) src - modified F90 source code and compiled version using debug flag
   b) src_O1 - compiled version using optimization flag 1
   c) verify
      i) fem - finite element-based problems
      ii) fpi - fpi-based problems
      iii) mpplrs - most probable point response surface-based problems
      iv) pbem - boundary element-based problems
      v) pfem - probabilistic finite element-based problems
      vi) risk - risk assessment-based problems
      vii) simfem - finite element sampling-based problems
      viii) sra - system risk assessment-based problems

The source code was compiled with the optimization flag -O1 in src_O1. The executable was verified using approximately 200 verification problems. The verification problems used are listed in Appendix C.

Appendix A - f90_params.f90 module

The `f90_params.f90` module is listed below. The routine contains the integer values for the upper case alphabet, numerals 0-9, and a few other relevant characters such as +, -, blank, and *. Subroutine `char_check` is used to verify that the integer values for the characters of whatever system on which NESSUS is running agree with the values used in the parameters. This routine is called from the `nessus.f` main routine. If any values do not agree, the program prints an error message and stops. It is not expected that this will occur, but it is useful to have a verification check.

Subroutine `char_from_int` is used as a convenient function to convert integers back to characters before printing.

```
!  module to define integer equivalents of characters.
!  values defined using parameters so that they can then
!  be used in data statements.  The transfer function is
!  used to ensure that the values agree with
!  the parameters for the current platform.  Routine char_check
!  should be called only once from the NESSUS main routine or
!  an initialization routine.

!  uses "transfer" intrinsic

   module f90_params

integer, PARAMETER :: AF90 = 1092624416, BF90 = 1109401632, &
               CF90 = 1126178848, DF90 = 1142956064,&
               EF90 = 1159733280, FF90 = 1176510496,&
               GF90 = 1193287712, HF90 = 1210064928,&
               IF90 = 1226842144, JF90 = 1243619360,&
               KF90 = 1260396576, LF90 = 1277173792,&
               MF90 = 1293951008, NF90 = 1310728224,&
               OF90 = 1327505440, PF90 = 1344282656,&
               QF90 = 1361059872, RF90 = 1377837088,&
               SF90 = 1394614304, TF90 = 1411391520,&
               UF90 = 1428168736, VF90 = 1444945952,&
               WF90 = 1461723168, XF90 = 1478500384,&
               YF90 = 1495277600, ZF90 = 1512054816

     integer, PARAMETER :: BLF90 = 538976288, N0F90 = 807411744,&
               N1F90 = 824188960, N2F90 = 840966176,&
               N3F90 = 857743392, N4F90 = 874520608,&
               N5F90 = 891297824, N6F90 = 908075040,&
               N7F90 = 924852256, N8F90 = 941629472,&
               N9F90 = 958406688, PLF90 = 723525664,&
               MNF90 = 757080096, EQF90 = 1025515552,&
               DECF90 =773857312, AMPF90 =639639584,&
               STF90 = 706748448, LPF90 = 673194016,&
               RPF90 = 689971232, CMF90 = 740302880
```

```
!       descriptions
!     AF90 = 'A'
!     BF90 = 'B'
!      etc.
!     BLF90 = BLANK
!     PLF90 = '+'
!     MNF90 = '-'
!     EQF90 = '='
!     DECF90 = '.'
!     AMPF90 = '&'
!     STF90 = '*'
!     LPF90 = '('
!     RPF90 = ')'
!     CMF90 = ','

    contains
      subroutine char_check
      implicit none

      integer :: ierr
      ierr = 0

!       check constants

      call char_compare('A   ',af90,ierr)
      call char_compare('B   ',bf90,ierr)
      call char_compare('C   ',cf90,ierr)
      call char_compare('D   ',df90,ierr)
      call char_compare('E   ',ef90,ierr)
      call char_compare('F   ',ff90,ierr)
      call char_compare('G   ',gf90,ierr)
      call char_compare('H   ',hf90,ierr)
      call char_compare('I   ',if90,ierr)
      call char_compare('J   ',jf90,ierr)
      call char_compare('K   ',kf90,ierr)
      call char_compare('L   ',lf90,ierr)
      call char_compare('M   ',mf90,ierr)
      call char_compare('N   ',nf90,ierr)
      call char_compare('O   ',of90,ierr)
      call char_compare('P   ',pf90,ierr)
      call char_compare('Q   ',qf90,ierr)
      call char_compare('R   ',rf90,ierr)
      call char_compare('S   ',sf90,ierr)
      call char_compare('T   ',tf90,ierr)
      call char_compare('U   ',uf90,ierr)
      call char_compare('V   ',vf90,ierr)
      call char_compare('W   ',wf90,ierr)
      call char_compare('X   ',xf90,ierr)
      call char_compare('Y   ',yf90,ierr)
      call char_compare('Z   ',zf90,ierr)
```

```
      call char_compare('    ',blf90,ierr)
      call char_compare('0   ',n0f90,ierr)
      call char_compare('1   ',n1f90,ierr)
      call char_compare('2   ',n2f90,ierr)
      call char_compare('3   ',n3f90,ierr)
      call char_compare('4   ',n4f90,ierr)
      call char_compare('5   ',n5f90,ierr)
      call char_compare('6   ',n6f90,ierr)
      call char_compare('7   ',n7f90,ierr)
      call char_compare('8   ',n8f90,ierr)
      call char_compare('9   ',n9f90,ierr)

      call char_compare('+   ',plf90,ierr)
      call char_compare('-   ',mnf90,ierr)
      call char_compare('=   ',eqf90,ierr)
      call char_compare('.   ',decf90,ierr)
      call char_compare('&   ',ampf90,ierr)
      call char_compare('*   ',stf90,ierr)
      call char_compare('(   ',lpf90,ierr)
      call char_compare(')   ',rpf90,ierr)
      call char_compare(',   ',cmf90,ierr)

      if (ierr /= 0) then
         write(*,*) 'Error in f90 parameters'
         write(*,*) 'Terminating NESSUS'
         stop
      end if

    end subroutine char_check

!-----------------------------------------------------------------------
    subroutine char_compare(chr,intval,ierr)
    implicit none

    character(len=4), intent(in) :: chr
    integer, intent (in) :: intval
    integer, intent(in out) :: ierr

      if (intval /= transfer(chr,intval)) then
         write(*,1) chr,intval, transfer(chr,intval)
         ierr = ierr + 1
      end if
1     format('Error, integer representation for character ', &
        a4,/,' value = ',i12,' not equal to parameter value = ',i12)
    end subroutine char_compare

!-----------------------------------------------------------------------
     function int_from_char(chr1) result(int)
!    routine determines the integer equivalent to the character
!    variable chr1

!    chr1 is character*1
```

```
!     therefore, 3 blanks are padded before the transfer
      integer :: int
      character(len=1) :: chr1
      character(len=4) :: chr4

      chr4 = chr1//'   '
      int = transfer(chr4,nopt)

!     write(*,*) '[int_char_set] chr4 = ',chr4,int_char_set
      return
      end function int_from_char

!-----------------------------------------------------------------------
      function char_from_int(int,n) result(chr4)

      integer, dimension(n) :: int

      character(len=4) :: tchr4
      character(len=4),dimension(n) :: chr4

      do i = 1,n
         chr4(i) = transfer(int(i),tchr4)
      end do
!     write(*,*) 'chr4 = ',chr4
      return
      end function char_from_int
   end module f90_params
```

Appendix B - Modified routines

bansf.f	-	change TITLE to ITITLE and use char_from_int to write as character
bem.f	-	removed Hollerith statements
betaic.f	-	removed Hollerith statements
betaln.f	-	removed Hollerith statements
blck01.f	-	made IDAT character*4
bounin.f	-	removed Hollerith statements.
cdf6.f	-	Add check for uniform distribution.
cdfglb.f	-	RWORK not dimensioned in FASTMC. Add RWORK to all calls to FASTMC and other calling subroutines.
cdfloc.f	-	RWORK not dimensioned in FASTMC. Add RWORK to all calls to FASTMC and other calling subroutines.
cfield.msc.f	-	variable card changed to cardn.
chi.f	-	Change convergence criteria from 1E-4 and 10 to 1E-8 and 50. Affects Harbitz sampling.
cintvl.f	-	RWORK not dimensioned in FASTMC. Add RWORK to all calls to FASTMC and other calling subroutines.
cnodel.f	-	removed Hollerith statements
compar.f	-	removed Hollerith statements, uses f90_parameters for initialization
dater.f	-	inserted FORTRAN 90 intrinsic for date
datin1.f	-	removed Hollerith statements
datin2.f	-	removed Hollerith statements
datin3.f	-	removed Hollerith statements, uses f90_parameters for initialization
datoh1.f	-	removed Hollerith statements, fixed output statements using variable A6
datou1.f	-	fixed output statements using variable A6, uses f90_parameters for initialization
datou4.f	-	removed Hollerith statements
decint.f	-	removed Hollerith statements, uses f90_parameters for initialization
decrea.f	-	removed Hollerith statements, uses f90_parameters for initialization
fastmc.f	-	RWORK not dimensioned in FASTMC. Add RWORK to all calls to FASTMC and other calling subroutines.
fcfout.f	-	removed Hollerith statements, changed NAME2, NAME3 to character
fem.f	-	removed Hollerith statements
finver.f	-	removed Hollerith statements
foloin.f	-	removed Hollerith statements

frontf.f	-	removed Hollerith statements
harbz2.f	-	fixed format statement past column 72
harmin.f	-	removed Hollerith statements, uses f90_parameters for initialization
header.f	-	removed Hollerith statements, added corrections to write out integer date variable
histo.f	-	removed Hollerith statements
hstout.f	-	change TITLE to ITITLE and use char_from_int to write as character
hypgeo.f	-	removed Hollerith statements
initfr.f	-	removed Hollerith statements
initi2.f	-	removed Hollerith statements
initin.f	-	removed Hollerith statements, uses f90_parameters for initialization
int_char_set.f	-	new function which sets integers based on character input
iopen.msc.f	-	corrected variable IRET which was used but not set
isreal.f	-	removed Hollerith statements, uses f90_parameters for initialization
jacobi.f	-	removed Hollerith statements
key.f	-	removed Hollerith statements, used TRANSFER function to write out input
llinp1.f	-	removed Hollerith statements, uses f90_parameters for initialization
llinp2.f	-	uses f90_parameters for initialization
lltim.f	-	removed Hollerith statements
llrvin.f	-	removed Hollerith statements, uses f90_parameters for initialization
matprt.f	-	modified print statement to print integers as alpha
models.f	-	modified argument to MAX
moniin.f	-	removed Hollerith statements, uses f90_parameters for initialization
monitr.f	-	removed Hollerith statements, changed NAME to character*4
mscpost.f	-	removed Hollerith statements
nessus.f	-	removed Hollerith statements, added case select to select NESSUS module
outdat.f	-	change TITLE to ITITLE and use char_from_int to write as character
page.f	-	removed Hollerith statements
page2.f	-	removed Hollerith statements, changed output of NAMVAR to character before printing
page2s.f	-	removed Hollerith statements, changed output of NAMVAR to character before printing
page3.f	-	removed Hollerith statements, changed output of NAMVAR to character before printing
page3s.f	-	removed Hollerith statements, changed output of NAMVAR to character before printing
pcfout.f	-	removed Hollerith statements, changed NAME2, NAME3 to character, adjusted format statements
perdef.f	-	removed Hollerith statements, uses f90_parameters for initialization
persiz.f	-	removed Hollerith statements, uses f90_parameters for initialization

pfem.msc.f	-	moved MFEMRV from data statement to executable
pmtrx.f	-	change TITLE to ITITLE and use char_from_int to write as character
pparti.f	-	removed Hollerith statements, changed A6 to character*6
preinp.f	-	removed Hollerith statements, uses f90_parameters for initialization
pretim.f	-	removed Hollerith statements, changed variable NAME to character before printing
prinin.f	-	removed Hollerith statements, uses f90_parameters for initialization
prino1.f	-	changed FORMS to character*8
prino2.f	-	changed FORMS to character*8
prino3.f	-	changed FORMS to character*8
prinou.f	-	removed Hollerith statements, initialize NAME1 and NAME2 with transfer function, changed FORMS, FORMS1 and FORMS2 to character
prinsu.f	-	removed Hollerith statements, initialize NAME with transfer function, changed FORMS, FORMS1 and FORMS2 to character
printm.f	-	removed Hollerith statements
prompt.f	-	uses command line argument, only routine different from HP
psdin.f	-	uses f90_params for initialization of NAME
quit.f	-	removed Hollerith statements, changed arguments to char*4 from char*6
r1mach.f	-	removed Hollerith statements
randin.f	-	uses f90_params for initialization of NAME2 and NAME3
rdpar.f	-	convert TITLE from integer to character
redext.msc.f	-	replaced bad logical statement, now is (.not.oldfil)
rednas.msc.f	-	removed variable IREDMSC, referenced but not set
redpfm.msc.f	-	added checks for valid FPI methods during a PFEM analysis
respon.f	-	comment out common PP
rottrn.f	-	Allow small negative eigenvalues when transforming from correlated variables to independent variables.
s88fmt.f	-	uses f90_params for initialization of IDIGIT
sbcin.f	-	removed Hollerith statements
select.f	-	uses f90_params for initialization of ICMMNT
setdef.f	-	removed Hollerith statements, uses f90_parameters for initialization
setup.f	-	removed Hollerith statements
simres.f	–	Add a close statement for scratch file.
sspev.f	-	removed Hollerith statements
string.f	-	removed Hollerith statements, read card as character variable then transfer to integer
subspc.f	-	removed Hollerith statements

syrsk2.f*	-	symbols referenced but not set (don't know how to fix)
sys.f	-	removed Hollerith statements, uses f90_parameters for initialization
sysana.f	-	removed Hollerith statements
timer.f*	-	must be updated with system dependent time routine
timout.f	-	removed Hollerith statements, changed arguments to character
tracin.f	-	removed Hollerith statements
traout.f	-	removed Hollerith statements
tyinin.f	-	removed Hollerith statements
ucfout.f	-	changed NAME2, NAME3 to character
uconv.f	-	Take absolute value of ZSTAR when making error check on Z for AMV+ analysis. Change ZERR_TOL from 0.2 to 0.01. Compare against zstar, which is the goal, for percent error not zfun, which is the latest iteration value.
verinc.f	-	moved initialization of IVERIN from data to executable
wrtpar.f	-	change TITLE to ITITLE and use char_from_int to write as character
xerprt.f	-	uses f90_params for initialization of F and G arrays
xerrwv.f	-	removed Hollerith statements
xersav.f	-	uses f90_params for initialization of F array
xfpi.f	-	RWORK not dimensioned in FASTMC. Add RWORK to all calls to FASTMC and other calling subroutines.
xtrrfm.f	-	fixed bad format statement
zlevel.f	-	RWORK not dimensioned in FASTMC. Add RWORK to all calls to FASTMC and other calling subroutines.

Appendix C -Verification problems

fem (59 files)

alstrip.dat
axicomp.dat
axihemi.dat
barbuck.dat
blister.dat
case1pre.dat
ccplate.dat
cfblade.dat
cspbuck.dat
cspgeom.dat
csppert.dat
cspsnap.dat
denotch.dat
disbeam.dat
disdyna.dat
dishole.dat
eptwcyl.dat
floydpv.dat
hpftp1s.dat
hpftp2s.dat
isohard.dat
kinhard.dat
lbeam3d.dat
lbeam3d_loub.dat
ldplate.dat
lpsbuck.dat
mixbeam.dat
mixdyna.dat
mixhole.dat
newarch.dat
newcap.dat
newwave.dat
nitwcyl.dat
ocbeam1.dat
ocbeam2.dat
ocbeam3.dat
onebrick.dat
pcplate.dat
phshell.dat
platepl.dat
prob1.dat
prob11.dat
prob1new.dat
prob2.dat
prob3.dat
prob4.dat
prob5.dat

prob6.dat
prob7.dat
prob8.dat
prob9.dat
rvfield.dat
sbeamf1.dat
sbeamf2.dat
sbeamf3.dat
stifeig.dat
stiff3d.dat
triplet.dat
twisted.dat
walkers.dat

fpi (41 files)

anne_amv.dat
anne_form.dat
cbeam_and.dat
cbeam_or.dat
column.fpi.m4.dat
column.fpi.m5.dat
column.fpi.m6.dat
column.fpi.m8.dat
column.fpi.m9.dat
lcfamv.dat
lcfmc.dat
norm100fpi.dat
qrplate_amv.dat
qrplate_fpi.dat
qrplate_mc.dat
tiltbeam_fpi.dat
tiltbeam_mc.dat
tpbend_ais2.dat
tpbend_amv.dat
tpbend_mc.dat
ueqn_ais2.dat
ueqn_amv.dat
ueqn_amv_cor.dat
ueqn_amv_plev.dat
ueqn_form_cor.dat
ueqn_fpi.dat
ueqn_mc.dat
ueqn_mv.dat
ueqn_sys.dat
ueqnsys.dat
ures_ais2.dat
ures_amv.dat
ures_amv_cor.dat
ures_fpi.dat
ures_mc.dat
ures_sys.dat
user_ais2.dat

user_amv.dat
user_fpi.dat
user_mc.dat
user_sys.dat

mpplrs (10 files)

p1-p2.dat
p1.dat
p2.dat
r-s.dat
r.dat
rsexact.dat
rsindep.dat
s.dat
tbexact.dat
tbindep.dat

pbem (44 files)

h0.dat
h1.dat
h10.dat
h11.dat
h12.dat
h13.dat
h14.dat
h15.dat
h16.dat
h17.dat
h18.dat
h19.dat
h2.dat
h20.dat
h21.dat
h22.dat
h23.dat
h24.dat
h25.dat
h26.dat
h27.dat
h28.dat
h29.dat
h30.dat
h31.dat
h34.dat
h35.dat
h36.dat
h37.dat
h38.dat
h39.dat
h40.dat
h41.dat

h42.dat
h45.dat
h47.dat
h48.dat
h49.dat
h6.dat
h7.dat
h8.dat
plate1.dat
plate2.dat
plate3.dat

pfem (32 files)

campbell_cf.dat
column.dat
column_aism.dat
columncf.dat
columncf_aism.dat
columncf_harb.dat
cor_opt0.dat
cor_opt1.dat
elplcy_cf.dat
elplcy_mccf.dat
elplcy_pfem.dat
fracture.dat
impp_p.dat
impp_z.dat
k1c.dat
k1c_fem.dat
lognormal.100.dat
normal.100.dat
qrplate.dat
qrplatecf.dat
test.dat
tiltbeam_pfem.dat
tiltbeam_pfemcf.dat
tpbend12.dat
ueqn.dat
ueqn_ais.dat
ueqn_amv.dat
ueqn_fpi.dat
ueqn_mc.dat
ueqn_mv.dat
ueqn_sys.dat
ueqnsys.dat

risk (2 files)

costtest.dat
perftest.dat

simfem (2 files)

qrplate.simfem.dat
simfem.h0.dat

sra (9 sra files)

eto_sys.dat
eto_sys_aismcf.dat
eto_syscf.dat
eto_sysimp.dat
sra_cbeam_and.dat
sra_cbeam_app.dat
sra_cbeam_imp.dat
sra_pframe.dat
sra_pframeg.dat